Adapting IRM/Linac Software

Implementing new projects

Mon, Jul 15, 2002

Many projects have been supported by Linac/IRM front-ends at Fermilab. This note describes some of the choices and steps required to implement a new project.

I/O interfaces

One of the earliest steps taken is determining what I/O interfaces will be used in a new project. For the case of an IRM, analog and digital interfaces are already available inside the crate. Eight bytes of digital I/O are available, configured as input or outbut by the byte. In addition, 64 channels of 16-bit analog data are included, in which the entire set of 64 channels is digitized automatically by hardware at a 1 KHz rate and stored into a circular buffer with room for 512 sets of such data. Additional IP boards can be added to support a set of 8 timers, a set of 8 fast digitizer channels that can be digitized at rates up to 800 KHz, or another set of 8 that can be digitized at rates up to 10 MHz. These are not the only choices, but they may be enough to get the discussion started.

After deciding on the I/O interfaces that will be used to acquire data, some thought may be given to the kinds of control available. The digital data can be controlled as bytes configured for outputs. Analog data can be supported by a set of 8 D/A channels, which are included with the usual 64 channel analog interface.

Characterization of what signals relate to what I/O interfaces is done via instructions entered into the nonvolatile Data Access Table, which describes how the data pool is to be updated each 15 Hz cycle. A large number of instruction types are available within the basic system software. Additional needs can be met by writing one or more local applications that update a portion of the data pool. The local applications are normally invoked at the end of the Data Access Table processing, when raw data from the I/O interfaces is freshly available. Once the data pool has been updated, all data looks the same—merely entries in analog and digital data tables.

Active data requests for which a reply is due on the current cycle are fulfilled from the data pool and replies delivered to the requesting nodes. This process is designed to operate efficiently, as the original data request was "compiled" into an internal representation during initial processing.

The next major step inside the system is to scan for alarm conditions of all analog and digital data for which alarm scanning is enabled, a job that typically requires less than 1 ms. Next, the local page application is given a chance to run. The entire process of data pool update, request fulfillment, and alarm scanning, begins again on the next 15 Hz cycle.

A description of each analog channel and digital bit is needed, so that data requests can be suitably fulfilled. For Acnet, this is done via the process of `DABBEL` entry, which enters the relevant device parameters into the central database. Following batch processing of `DABBEL` commands, a special post-processing program downloads the information to the front-end so that the appropriate tables can be set up.

Nonvolatile system tables relating to characterization

The Data Access Table contains a list of instructions that define how to update the data pool every 15 Hz cycle. Some instructions are built-in as part of the basic system code.

Additional abilities can be created via local application programs, which are separately compiled and downloaded from the basic system. The only external functions referenced by a local application are within the system code; local applications do not directly reference each other. As a result, local applications can be loaded or unloaded, activated or deactivated in any order. They serve as modular extensions of the system code to satisfy specific needs for data pool update, closed loop algorithms, support of network protocols, and diagnostics.

The Analog Descriptor Table defines fixed attributes of analog channels, including device specifications for settable channels and related digital status and control bits. Its entries are downloaded as a result of the automatic post-processing following `DABBEL` database entry. The Analog Data Table can be considered the integer analog data pool, as its entries contain dynamic attributes for each analog channel, including the reading, setting, nominal, tolerance and alarm flags. A parallel table for raw floating point analog channels is also available. Fast digitizer parameters are defined by entries in the Channel Information Table, or `CINFO`. The Binary Address Table defines how each byte of digital data is to be read or set. The Binary Descriptor Table includes descriptive text for each digital bit. The Binary Alarm Table, which includes the digital alarm flags and trip counts, along with the Binary Byte Table, which holds the most recent byte readings, comprise the digital data pool. The Combined Status Table, or `CSTAT`, specifies how digital status words are constructed from the basic raw binary data in the digital data pool.

Entries in the Data Access, `CINFO`, `CSTAT`, and Binary Address tables are set via memory accesses to those areas of memory where the tables reside, usually by using the Memory Dump page application. Other page applications assist in entering parameters for the Binary Descriptor and Binary Alarm tables. A special utility is available that runs on a PC to help initialize Data Access Table entries.

Local application parameters are set via a special page application that acts as the common user interface for all local application instances within a front-end.

Note that since any front-end can access any other front-end, a page application may run in any front-end to help configure any other. Even when “little console” hardware is unavailable, a so-called “Page G” emulator can run on various platforms to gain access to and operate any page application in any front-end.